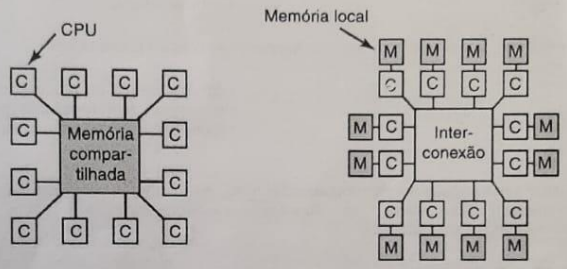
Eduardo Pereira

Universidade de Caxias do Sul
Área do Conhecimento de Ciências Exatas e Engenharias
CIC4004A - Programação Concorrente, Paralela e Distribuída
Professor: André Luis Martinotto
Primeira Avaliação

QUESTÃO 1) (1,5 PONTOS) Na figura abaixo, retirada do livro "Sistemas Operacioanis Modernos" de Andrew Tanenbaum, tem-se diferentes exemplos de arquiteturas paralelas.

a) Identifique a qual categoria da taxonomia de Flynn pertencem as arquiteturas abaixo. Justique sua resposta.

b) Quais dessas arquiteturas paralelas podem ser classificadas com um sistema distribuído? Por quê?



QUESTÃO 2) (1,0 PONTOS) Considere uma situação em que sensores de IoT capturam dados de temperatura em tempo real e enviam para um servidor que controla a temperatura de um armazém. Se os dados não chegarem ao servidor dentro de um limite de tempo, eles são descartados para evitar informações desatualizadas na gestão de estoques sensíveis à temperatura. Qual protocolo de comunicação, TCP ou UDP, seria mais apropriado para garantir a eficácia na transmissão dos dados e atender às necessidades específicas dessa aplicação? Justifique a sua escolha com base nas características dos dois protocolos.

QUESTÃO 3) (1,0 PONTO) Protocol Buffers (Protobuf) é um formato de dados binário, enquanto o JSON (JavaScript Object Notation) é um formato textual. Ambos são comumente empregados na comunicação em sistemas distribuídos. Ao analisar esses formatos, compare as características de Protocol Buffers e JSON. Quais são as vantagens de cada um e de que forma essa escolha pode impactar o desempenho em ambientes distribuídos?

QUESTÃO 4) (1,5 PONTO) Um programa que realiza um processamento intensivo leva 100 segundos para ser executado em um único núcleo de CPU. Após a implementação de uma versão paralela, o mesmo programa é executado em 4 núcleos de CPU e leva 30 segundos para ser concluído.

a) Calcule o speedup obtido com a execução do programa em paralelo.

b) Calcule a eficiência do programa em relação ao número de núcleos utilizados.

QUESTÃO 5) (2,5 PONTOS) Você foi encarregado de desenvolver uma uma aplicação cliente/servidor utilizando sockets TCP para transferência de arquivos. Nessa aplicação o processo cliente lê o nome de um arquivo e faz uma requisição desse arquivo ao processo servidor. O processo servidor recebe essa requisição e efetua a transferência do arquivo ao processo cliente.

Deseja-se que o servidor possua a capacidade de atender múltiplos clientes simultaneamente, isto é, no caso de uma requisição de um cliente por um arquivo grande os demais clientes não deverão ficar esperando para serem atendidos.

Para o desenvolvimento da aplicação você deve utilizar uma biblioteca de sockets que apresenta as seguintes primitivas:

sock = socket(Protocolo):bind(socket, porta):	Criação de um *socket*. Protocolo TCP ou UDP.
bind(socket, porta):	Estabelece uma porta ao *socket*.
listen(socket, quantos):	Definição do tamanho da lista de requisições.
connect (socket, ip, porta):	Requisição de uma conexão.
new_sock = accept (socket):	Espera por uma conexão. Retorna um *socket* que será usado para a comunicação.
send (socket, dados):	Envia uma mensagem com o conteúdo de dados;
recv(socket, dados):	Recebe uma mensagem e armazena em dados;
close(socket):	Fecha a conexão;

Para permitir o atendimento de múltiplos clientes simultaneamente utilize a primitiva fork:

int fork():	cria uma cópia do processo original. A chamada de sistema fork retorna o pid do filho para o processo pai e 0 para o processo filho.

OBSERVAÇÃO: Utilize uma linguagem de programação real. As funções para transferência de arquivos não precisam ser implementadas utilize as funções *enviarArquivo(socket, string nomeArquivo)* e *receberArquivo(socket, string nomeArquivo)*

QUESTÃO 6) (2,5 PONTOS) O trecho de código abaixo calcula a variância e o desvio padrão para um vetor amostra[N].

```
soma = 0;
for (i=0;i<N;i++){
        soma += amostra[i];
}
media = soma/N;

soma = 0;
for (i=0;i<N;i++){
        soma += (amostra[i]-media) * (amostra[i]-media);
}

variancia = soma/(N-1);
desvio = sqrt(variancia);
```

Utilizando como suporte uma biblioteca de troca de mensagens que segue o modelo SPMD (único programa trabalhando sobre múltiplos dados) e que possua as primitivas abaixo e uma linguagem de programação real, reescreva o programa para que o cálculo seja efetuado em paralelo por *P* processos.

Primitivas disponíveis:

int rank():	Retorna o identificador de cada processo (de 0 a N-1);
int size():	Retorna o número de processos;
send(<var>,<id>):	Envia uma mensagem contida em <var> para a tarefa identificada por <id> .
receive(<var>,<id>):	Recebe do processo <id> e armazena no endereço <var>;

Eduardo Ferreira

1) a - Ambas são MIMD, visto que são arquiteturas multiprocessadas, e assim possuem múltiplas instruções para múltiplos dados.

b - Apesar a segunda arquitetura pode ser considerada distribuída, visto que ela possui recursos não compartilhados, isto é, cada CPU possui uma memória local, o que exige conexão entre essas máquinas. Já a primeira arquitetura é multiprocessada, mas todos processadores têm acesso à uma mesma memória compartilhada, o que a desqualifica como distribuída.

2) O protocolo UDP seria mais adequado. Além de ser mais rápido, o que permite maior agilidade ao lidar com dermineis na temperatura, ~~o protocolo UDP descarta pacotes que não não entregues~~, coincidindo com o requisito do sistema de não enviar informações desatualizadas. Se o protocolo TCP fosse usado, ele insistiria nessa entrega.

3) O protobuff, por usar dados binários, é mais leve e flexível, porém mais difícil de tratar, o que demanda mais processamento. Já o JSON, por ser um formato textual, ocupa mais espaço e, portanto, pode atrasar a transmissão. No entanto, é facilmente manipulado e compreendido. Assim sendo, protobuff é ideal para casos com menor disponibilidade de comunicação e maior processamento no servidor, enquanto JSON é ideal para redes robustas e computadores com alto desempenho.

4)

núcleos	speedup	speedup - esperado	eficiência
1	1	1	100% ou 1
4	3,33	4	83,25% ou 0,8325

↳a ↳b

5)

cliente:

```
scanf("%s", &arquivo);
sock = socket(TCP);
connect(sock, ip, porta);
send(sock, arquivo);
receberArquivo(sock, arquivo);
close(sock);
```

servidor:

```
#define N 5  #define porta 6000
sock = socket(TCP);
bind(sock, porta);
listen(sock, N);
ns = accept(sock);
int pid = fork();
if(pid == 0){ // filho
    receive(ns, arquivo);
    enviarArquivo(ns, arquivo);
    close(ns);
    exit(0); // fecha processo filho
}
close(sock);
```

6)

```
p = size()
soma = 0
id = rank()
for(i=id; i<N; i+=p)
    soma += amostra[i];
if(id == 0){
    for(i=1; i<p; i++){
        receive(aux, i)
        soma += aux; }
    media = soma/N
    for(i=1; i<p; i++)
        send(media, i)
} else {
    send(soma, 0)
    receive(media, 0)
}
cont...
```

cont...

```
soma = 0
for(i=id; i<N; i+=p){
    soma += (amostra[i] - media) *
            (amostra[i] - media)
if(id == 0){
    for(i=1; i<p; i++){
        receive(aux, i)
        soma += aux }
    variancia = soma / (N-1)
    desvio = sqrt(variancia)
} else {
    send(soma, 0)
}
```